# Formal Security Analysis with Interacting State Machines

David von Oheimb and Volkmar Lotz
Siemens AG, Corporate Technology, D-81730 Munich
{David.von.Oheimb|Volkmar.Lotz}@siemens.com

# Motivation

industrial IT system development

rigorous analysis of security properties

evaluation according to ITSEC and CC

formal security modeling and verification

machine assistance for checks and presentation

# Goals

**Expressiveness:** state transformation, concurrency and message passing

**Flexibility:** adaptation and extension

**Simplicity:** minimal expertise and time

**Graphical capabilities:** overview and intuition

**Maturity of the semantics:** refinement and compositionality

**Availability of tools:** open-source preferred

# Related Work

**IOA Language and Toolset** for specification, simulation, theorem proving, model checking, and code generation

**I/O Automata (IOAs)** model for asynchronous distributed computation

- ⊕ well-developed meta theory
- ⊕ theorem proving and model checking support
- ⊖ low-level interaction scheme (singleton, unbuffered input/output)

**Focus** based on synchronized stream-processing functions

**Other approaches** combine state-oriented and process-oriented description

# Isabelle/HOL

*Isabelle* generic theorem prover

Instantiation for *Higher-Order Logic (HOL)*

Predicate logic + simply-typed $\lambda$-calculus

- $\oplus$ automatic typing
- $\ominus$ lack of dependent types
- $\oplus$ well-documented, well-supported, freely available
- $\oplus$ very flexible and mature verification environment

# AutoFocus

Graphical specification and simulation tool for distributed systems

Back-ends: code generators, model checkers

partially commercial: *Validas* AG

*System Structure Diagrams (SSDs)*, *State Transition Diagrams (STDs)*, *Data Type Definitions (DTDs)*, *Extended Event Traces (EETs)*

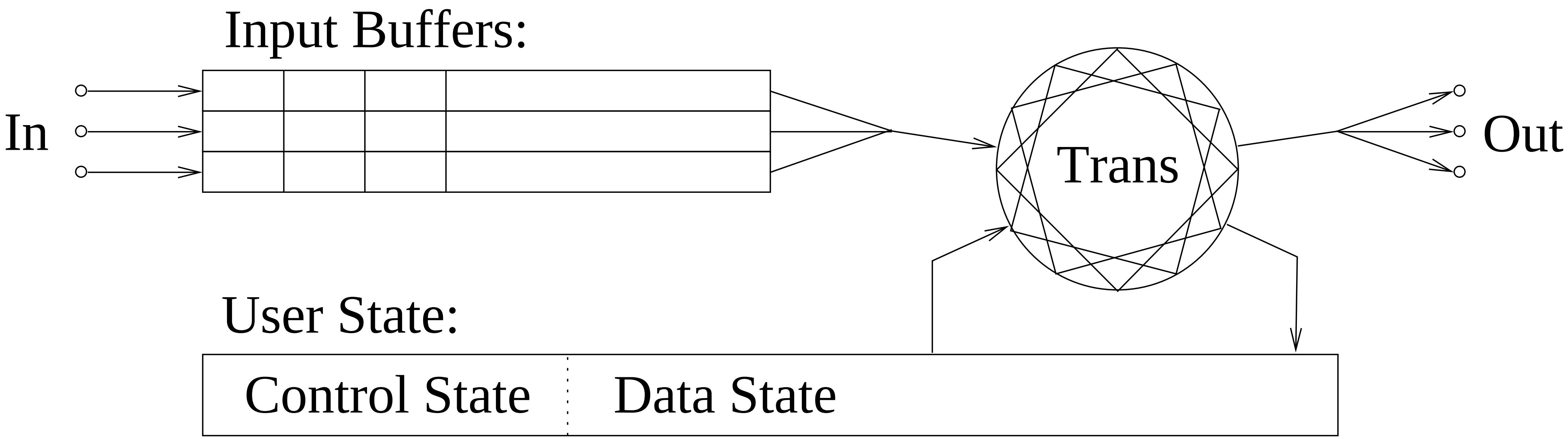
⊕ graphical design and presentation

⊖ limited expressiveness (type system, underspecification)

⊖ clock-synchronous semantics

# Interacting State Machines (ISMs)

Automaton with (nondeterministic) state transitions +
Buffered i/o simultaneously on multiple connections
Operational (executable) and/or axiomatic (property-oriented) style




Infinite executions (and thus also fairness) not considered

# Tool Representation

AutoFocus drawing $\longrightarrow$ Quest file $\xrightarrow{\text{Conv}_1}$ Isabelle theory file

Within Isabelle: **ism** sections $\xrightarrow{\text{Conv}_2}$ Standard HOL defintions

## AutoFocus

Syntactic perspective

Type and consistency checks

Graphical documentation

## Isabelle

Semantic perspective

Validation and correctness proofs (interactive)

Textual documentation

# LKW Model for the Infineon SLE66

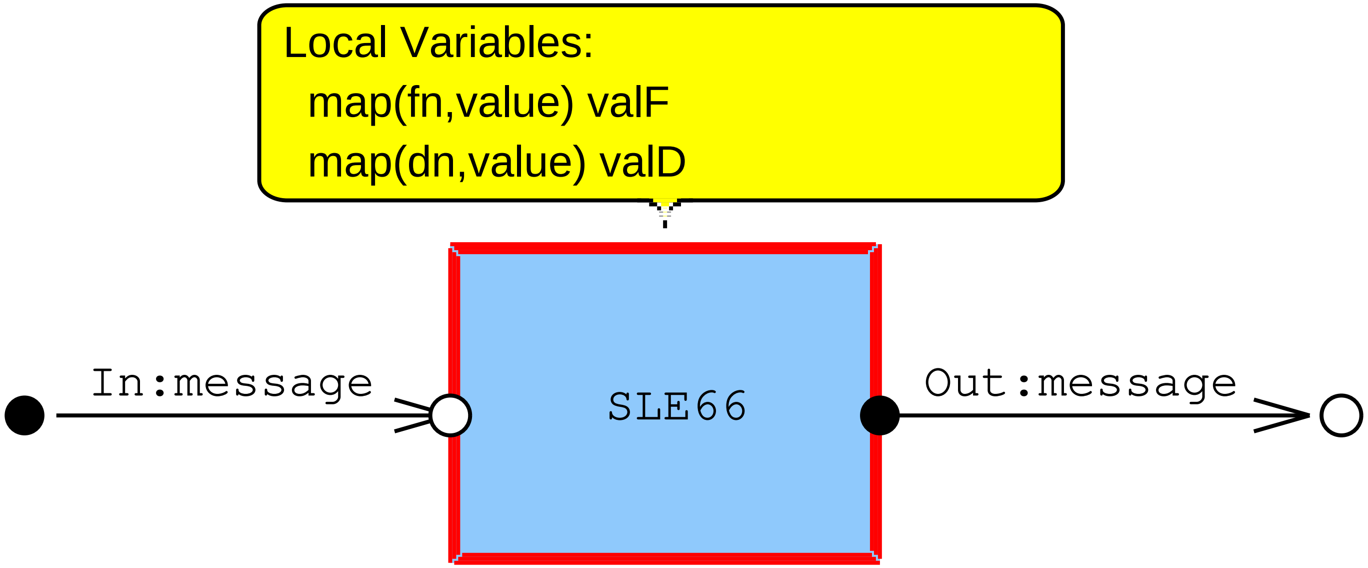
One of first formal models for security properties of hardware
Security evaluation for SLE66 on ITSEC E4 / CC EAL5 level

Abstract system model based on an ad-hoc automaton formalism
Formalization of security requirements, correctness proof

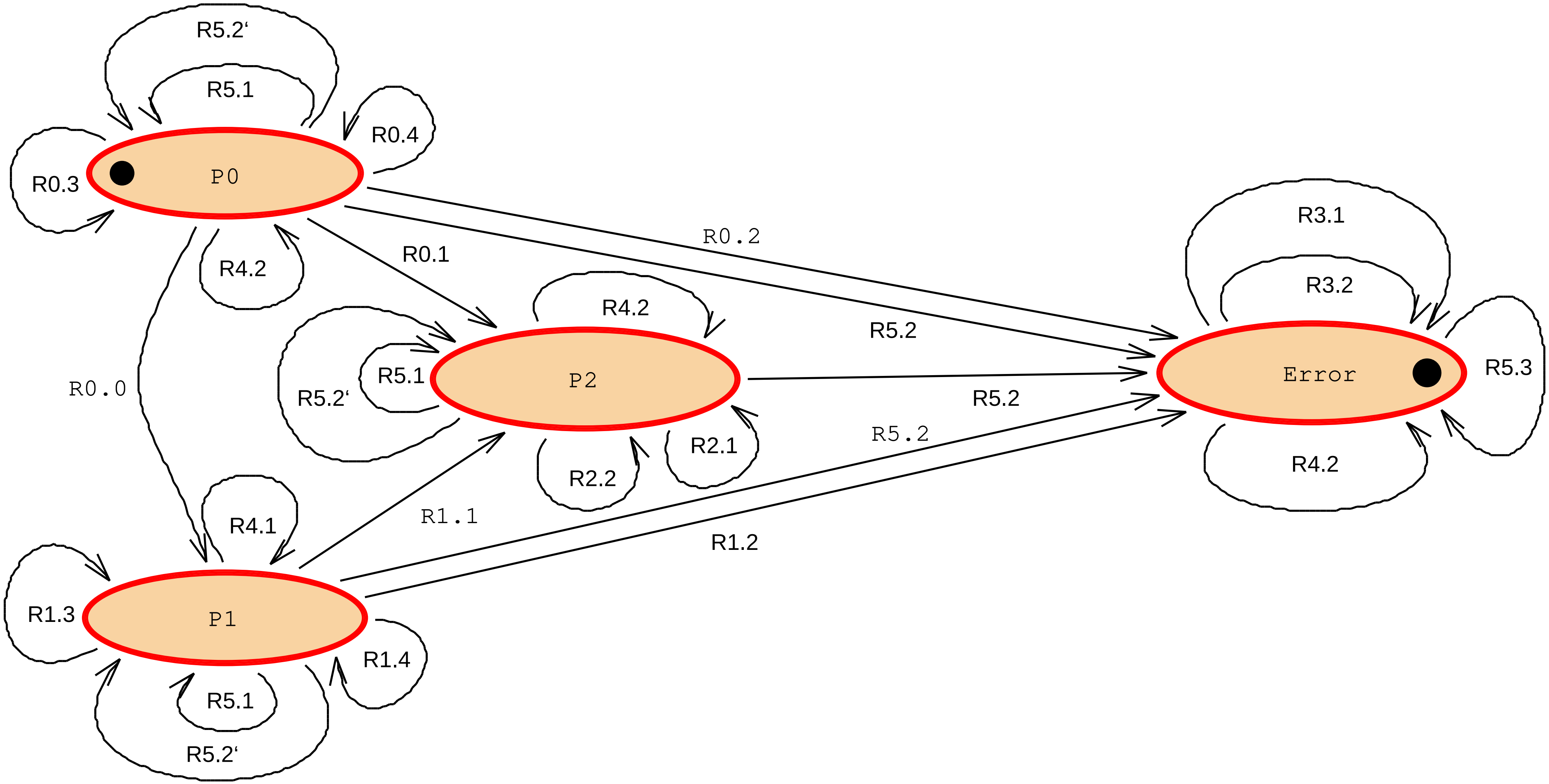
Minor syntactical, typographical and semantical slips
Type errors, missing assumptions, incomplete proofs

$\Rightarrow$ Use of Isabelle + automata formalism based on IOAs

# LKW Model: System Architecture



# LKW Model: State Transitions



# LKW Model: State etc.

Object names:

**datatype** *on = F fn | D dn*

Security-relevant objects ⇒ secrecy and integrity

$F\_Sec = F\_PSec \uplus F\_ASec$

Phases (control states):

**datatype** *ph = P0 | P1 | P2 | Error*

Data state:

**record** *data =*
  *valF :: "fn ⇝ val"*
  *valD :: "dn ⇝ val"*

Initial data state (underspecification):

**const** *s0 :: data*

Connection names:

**datatype** *interface = In | Out*

input and output messages:

**datatype** *message =*
  *Exec sb fn | Load sb fn val | Spy on*
*| Val val | Ok | No*

# LKW Model: ISM section, Rule 0.0

```
ism SLE66 =
  connections interface
    inputs    "{In}"
    outputs   "{Out}"
    messages  message
  state
    control P0 :: ph
    data    s0 :: data
  transitions

R0.0: P0 -> P1
      in   In  "[Exec Pmf f]"
      out  Out "[Ok]"
      pre  "f ∈ fct s∩FTest0; test f s"
      eff  "valF := valF s⌊-FTest0"
```

Typical: Both (single) input and (single) output

# LKW Model: Rule 5.2, 4.1

```
R5.2: ph -> Error
      in   In  "[Spy oname]"
      out  Out v
      pre  "ph≠Error; oname∈Sec;
             v∈{[],[Val (the (val s oname))]}"
      eff  "valF := fs, valD := ds"
```

Generic transition

Two sorts of nondeterminism

```
R4.1: P1 -> P1
      in   In  "[Load Pmf f v]"
      out  Out "[Ok]"
      pre  "f ∈ F_NSec ∪ (F_ASec - fct s)"
      eff  "valF := valF s(f↦v)"
```

*Load* operation also for application security domain

# Properties

Security objectives: constraints on the system behavior

Abstract specification requires a few axioms, e.g.

*Axiom1:* $"f \in fct\ s \cap F\_Sec \Longrightarrow valF\ (change\ f\ s) \lfloor F\_Sec = valF\ s \lfloor F\_Sec"$

...including slight changes according to new rule `R4.1`, also within...

**theorem** *FS021:* $"[\![((ib,(ph,s)),(p,ib',(ph',s'))) \in Trans;\ ph' \neq Error;\ g \in fct\ s \cap fct\ s' \cap F\_Sec]\!] \Longrightarrow valF\ s'\ g = valF\ s\ g"$

Isabelle proofs: just a few steps, 50% automatic

# Detected Omissions

**Axiom (for FSO1):** If a function object may be referenced in two (different) ways and one of them declares the function to be security relevant, the other has to do the same.

**Invariant (for FSO5):** Accessibility of test functions

**constdefs**

$no\_FTest\_invariant :: "state \Rightarrow bool"$

$"no\_FTest\_invariant \equiv \lambda(ph,s).$

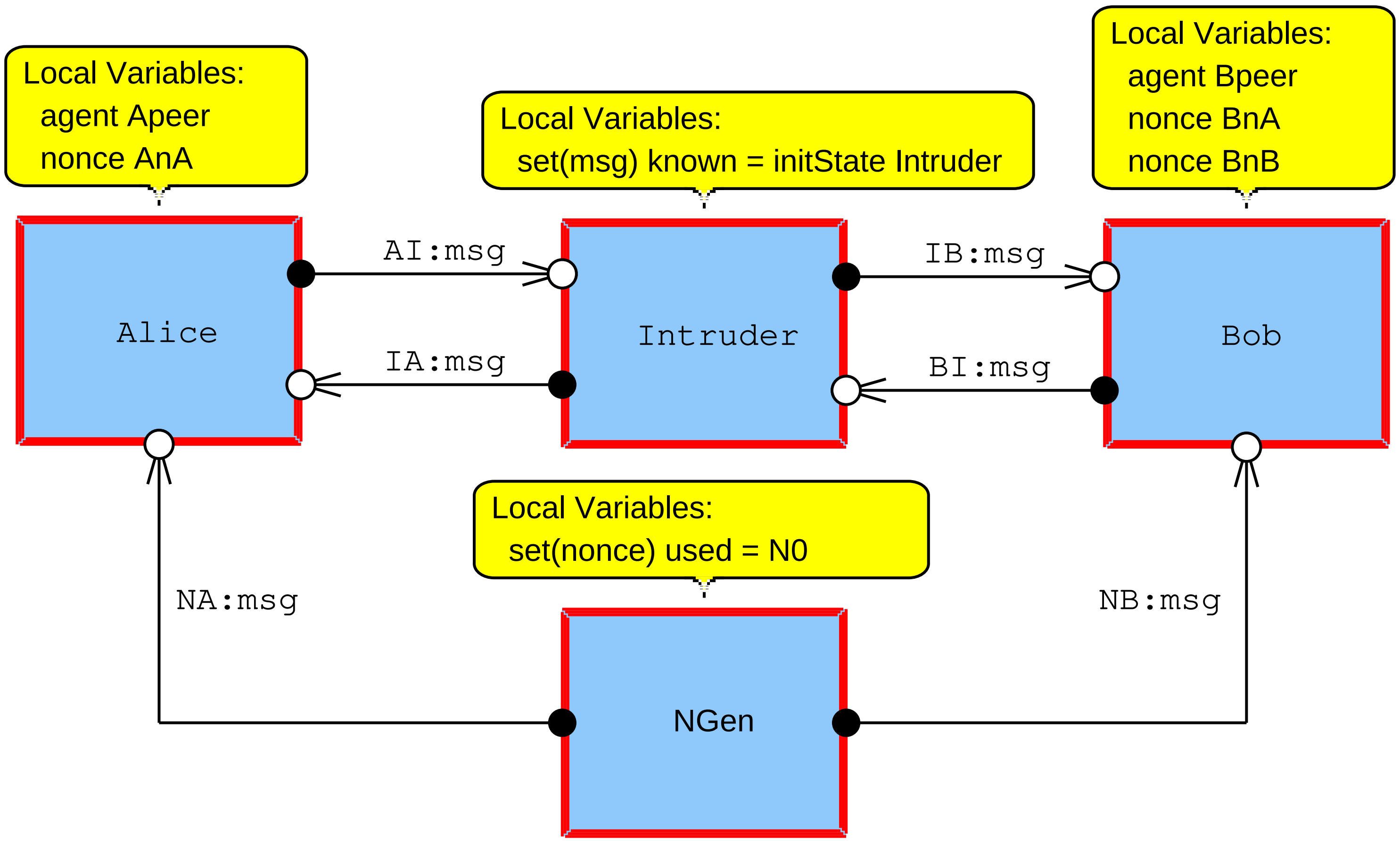
$\forall f \in fct\ s.\ (ph = P1 \longrightarrow f \notin FTest0) \wedge (ph = P2 \longrightarrow f \notin FTest)"$

**theorem** $FSO5$: $"[\![((ib,(\_,s)),(p,\_,(\_,s'))) \in Trans;\ ib\ In = Exec\ sb\ f\#r;$

$f \in FTest]\!] \Longrightarrow sb = Pmf \vee s' = s \wedge p\ Out = [No]"$

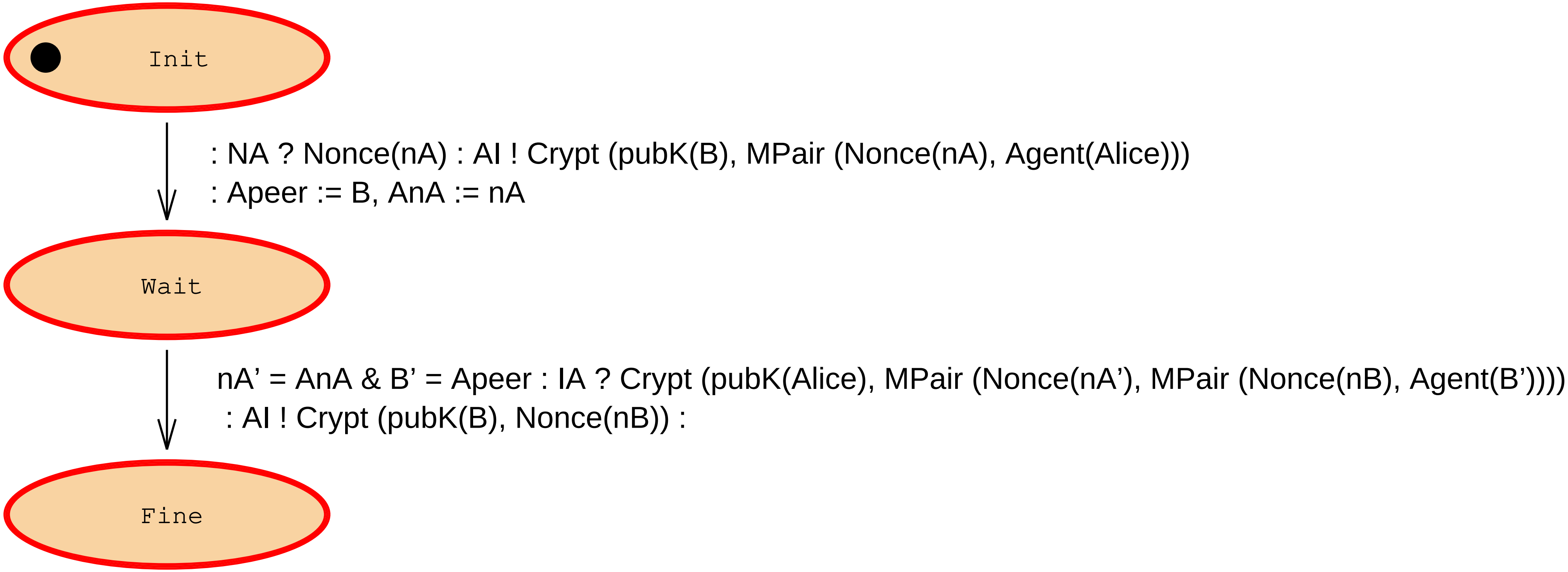
# Needham-Schroeder Public-Key Protocol

*NSL* as benchmark for security in distributed systems



# NSL State Transition Diagrams: `Alice`

Transition format: `guard : inputs : outputs : assignments`

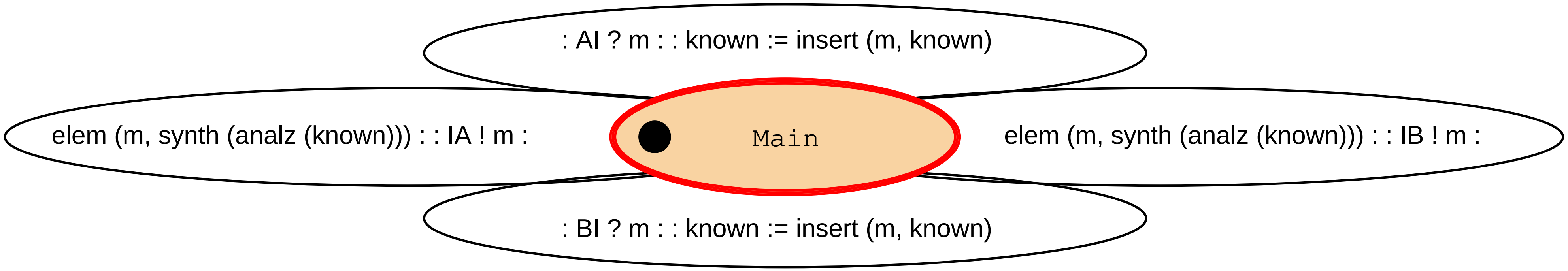



Multiple instances for interleaved protocol runs

# NSL State Transition Diagrams: others

`Bob`: similar to `Alice`

`Intruder`: centered around its current knowledge




`NGen`: similar, but no input

N.B. Agents have state:

- denoting current expectations and knowledge
- fixing order of protocol steps

# NSL Formalization

Formalization based on Paulson's "inductive approach",
tailored to semi-automated verification of cryptographic protocols

⊕ high automation due to abstraction to event traces

⊖ models and properties a bit cumbersome:
state is implicit, but often referred to
by repeating event history, adding auxiliary events

# NSL Model: Bob

```
ism Bob =
   connections channel
      inputs    "{NB,IB}"
      outputs   "{BI}"
      messages  msg
   state
      control Idle :: B_control
      data    B0   :: B_data
   transitions
 Resp: Idle -> Resp
       in   NB "[Nonce nB]",
            IB "[Crypt (pubK Bob)
           {|Nonce nA, Agent A|}]"
       out  BI "[Crypt (pubK A)
           {|Nonce nA, Nonce nB, Agent Bob|}]"
       eff  "Bpeer := A, BnA := nA, BnB := nB"
 Ack': Resp -> Conn
       pre  "nB' = BnB s"
       in   IB "[Crypt (pubK Bob) (Nonce nB')]"
```

Transition Resp: two inputs, one output

Transition Ack': precondition made explicit

# NSL Model: NGen

Freshness of nonces is intricate

Paulson (semantical): no appearance in event history

Does a local criterion suffice?

New solution: component makes generation explicit

```
ism NGen =
   connections channel
     inputs     "{}"
     outputs    "{NA,NB}"
     messages   msg
   state
     control "()" :: unit
     data    NO   :: "nonce set"
   transitions
  Cackle: "()" -> "()"
       pre    "ch ∈ {NA, NB}; n ∉ used s"
       out    ch "[Nonce n]"
       eff    "used := insert n (used s)"
```

Output connection non-deterministically selected

# NSL: Properties

...expressed with reference to agent state and message traffic

Authentication of `Alice` to `Bob` (even session agreement!)

$$\llbracket \mathit{Alice} \notin \mathit{bad};\ \mathit{Bob} \notin \mathit{bad};\ ((b,\ s),\ ts) \in \mathit{ISM\_runs\ System};$$
$$\exists nB.\ \mathit{Bob\_state}\ s = (\mathit{Conn}, (\!| \mathit{Bpeer} = \mathit{Alice},\ \mathit{BnA} = nA,\ \mathit{BnB} = nB |\!))\rrbracket \Longrightarrow$$
$$\exists t \in \mathit{set}\ ts.\ \mathit{Alice\_state}\ (\mathit{state'}\ t) = (\mathit{Wait}, (\!| \mathit{Apeer} = \mathit{Bob},\ \mathit{AnA} = nA |\!))$$

Paulson's formulation: referring to all messages of the protocol

$$\llbracket A \notin \mathit{bad};\quad B \notin \mathit{bad};\quad \mathit{evs} \in \mathit{ns\_public};$$
$$\mathit{Crypt}\ (\mathit{pubK}\ B)\ (\mathit{Nonce}\ NB) \in \mathit{parts}\ (\mathit{spies}\ \mathit{evs});$$
$$\mathit{Says}\ B\ A\ (\mathit{Crypt}\ (\mathit{pubK}\ A)\ \{\!|\mathit{Nonce}\ NA, \mathit{Nonce}\ NB, \mathit{Agent}\ B|\!\}) \in \mathit{set}\ \mathit{evs}$$
$$\rrbracket \Longrightarrow$$
$$\mathit{Says}\ A\ B\ (\mathit{Crypt}\ (\mathit{pubK}\ B)\ \{\!|\mathit{Nonce}\ NA, \mathit{Agent}\ A|\!\}) \in \mathit{set}\ \mathit{evs}$$

Proofs along the lines of Paulson, similar unicity and secrecy lemmas

Extra detail $\Rightarrow$ more painful, additional lemmas on intermediate states

# Conclusion: Concepts

ISMs as high-level IOAs
with additional structure and communication facilities
Support for relating input and output actions is important

ISM semantics expressable in terms of IOA semantics
inheriting mature semantic concepts: compositionality and refinement
Proof support inherited from Isabelle/IOA implementation

Further work:
Proof support directly on ISM level,
Specification using temporal logic,
Simulation support and test case generation

# Conclusion: Application

Machine checking reduces errors and omissions
like hidden assumptions and sloppy argumentation

Graphical representation in AutoFocus
suits design and documentation

Applicable to a variety of security analysis tasks:
High-level security modeling and requirements analysis
Lowel-level analysis of distributed systems like crypto protocols

$\Rightarrow$ ISM approach: good support for formal security analysis industry